НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
Олександр ПАВЛОВ
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-застосування для ідентифікації користувача за голосом*

Виконав: студент IV курсу, групи *ІП-63 Опанасенко Ярослав Павлович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *Старший викладач Ковтунець Олесь Володимирович* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доцент каф. ТК, к.т.н., доцент Віктор ПАСЬКО
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Опанасенку Ярославу Павловичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Веб-застосування для ідентифікації користувача за голосом*»

керівник проєкту *Ковтунець Олесь Володимирович, старший викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Аналіз якості та тестування програмного забезпечення
4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Схема бази даних

2) Схема структурна класів програмного забезпечення

3) Креслення вигляду екранних форм

4) Схема структурна бізнес-процесів

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*19.03.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*26.03.2020*	
3.	*Постановка та формалізація задачі*	*26.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*02.04.2020*	
5.	*Алгоритмізація задачі*	*02.04.2020*	
6.	*Моделювання програмного забезпечення*	*09.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*16.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*23.04.2020*	
9.	*Розробка програмного забезпечення*	*30.04.2020*	
10.	*Налагодження програми*	*07.05.2020*	
11.	*Виконання графічних документів*	*14.05.2020*	
12.	*Оформлення пояснювальної записки*	*21.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*03.05.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Ярослав ОПАНАСЕНКО
(підпис)

Керівник _____ Олесь КОВТУНЕЦЬ
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6322.045420.02.81	Пояснювальна записка	53	
3	А4	КПІ.ІП-6322.045420.03.91	Технічне завдання	12	
4	А4	КПІ.ІП-6322.045420.04.51	Програма та методика тестування	6	
5	А4	КПІ.ІП-6322.045420.05.34	Керівництво користувача	5	
6	А4	КПІ.ІП-6322.045420.06.13	Опис програми	17	
7	А3	КПІ.ІП-6322.045420.07.99.СБД	Схема бази даних	1	
8	А3	КПІ.ІП-6322.045420.07.99.СС	Схема структурна класів програмного забезпечення	1	
9	А3	КПІ.ІП-6322.045420.07.99.КЕ	Креслення вигляду екранних форм	1	
10	А3	КПІ.ІП-6322.045420.07.99.СС	Схема структурна бізнес процесів	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6322.045420.00.90		
Розробн.		Опанасенко Я.П.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 19 таблиць та 6 джерел.

Об’єкт дослідження: веб-застосування для ідентифікації користувача за голосом.

Мета дипломного проекту: розробити систему для ідентифікації користувача за голосом.

У першому розділі проведено аналіз відомих технічних рішень і розроблені вимоги до програмного забезпечення.

У другому було розроблено архітектуру веб-застосування. Було розглянуто основні методи реалізації та вибрано основний.

У третьому розділі проведено тестування даного веб додатку за розробленим планом тестування. Описано процес тестування.

У четвертому розділі описано розгортання та впровадження вебзастосування.

У додатках наведено: опис програми, схема структурна класів програмного забезпечення, схема структурна послідовності виконання.

КЛЮЧОВІ СЛОВА: ІДЕНТИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА, ВЕРИФІКАЦІЯ, ГОЛОСОВІ ЗВ’ЯЗКИ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 19 tables and 6 sources.

Object of research: a web application for user identification by voice.

The purpose of the diploma project: to develop a system for user identification by voice.

In the first section the analysis of the known technical decisions is carried out and requirements to the software are developed.

The second developed a web application architecture. The main methods of implementation were considered and the main one was chosen.

In the third section, we tested this web application according to the developed testing plan. The testing process is described.

The fourth section describes the deployment and implementation of a web application.

The appendices contain: description of the program, scheme of structural classes of software, scheme of structural sequence of execution.

KEY WORDS: IDENTIFICATION, NEURAL NETWORK, VERIFICATION, VOICE COMMUNICATIONS.

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування для ідентифікації користувача за голосом_____
_____
_____

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

Адміністратор – людина, що відповідає за функціонування системи та створення бази даних.

Користувач – людина, що проходить ідентифікацію

Біометрія – сукупність автоматизованих методів і засобів ідентифікації людини, заснованих на її фізіологічній або поведінковій характеристиці.

Кібер атака – дії кібер-зловмисників (кракерів) або шкідливих програм, які спрямовані на захоплення інформаційних даних віддаленого комп'ютера.

Голосові зв'язки – еластичні утворення, розташовані в середній частині гортані.

ДКЧП – довга короткочасна пам’ять (LSTM), один із видів рекурентних нейроних мереж.

VAD – voice activity detection, виявлення голосової активності.

РНМ – рекурентна нейронна мережа.

MFCC – mel-frequency cepstral coefficients, мел-частотні кепстральні коефіцієнти.

STFT – short-time Fourier transform, віконне перетворення Фур’є, один із різновидів перетворень. Фур’є.

DTFT – discrete-time Fourier transform, дискретне часове перетворення Фур’є, один із різновидів перетворень. Фур’є.

Шумозаглушна система – набір алгоритмів, спрямований на прибрання з аудіофайлу стороннього шуму.

Кракер – комп’ютерний злоумисник, дії якого спрямовані на неправомірне здобуття інформації, пошкодження даних або нанесення шкоди деякій комп’ютерній системі.

Верифікація – доказ того, що вірогідний факт або твердження є істинними.

КПП – контрольно пропускний пункт.

ВСТУП

Ні для кого не секрет, що в наш час комп'ютерні технології стали невід'ємною частиною життя кожної людини. Кожного дня все більше і більше приватих, державних та військових установ впроваждують ІТ-технології у структуру своєї роботи. Тому з впевненістю можна казати, що одним з головних питань до програм, які використовують дані структури є обмеження доступу користувачів. За останні два десятиріччя кракери досягли чималих результатів у мистецтві взлому, перехопленню та підробці інформації. Все частіше лунають новини про зломи баз данних, оприлюднення особистої інформації корисувачів та успішні хакерські атаки. Від таких атак лише за два останні роки світові компанії понесли збитки на сумму більше трьох трильйонів американських доларів. Щоб захистити приватні данні користувачів різних систем, методи захисту інформації повинні бути завжди на крок попереду. Стандартні методи захисту на кшталт паролів, смс-верифікацій та верифікацій за допомогою електронної пошти не працюють. Все частіше такі прості системи захисту доповнюються або змінюються біометричними системами ідентифікації. Термін біометрія означає вимір фізіологічних або анатомічних параметрів людини. Якщо звичайний пароль або смс повідомлення можна вкрасти, підібрати або перехопити, то обдурити біометричну систему вкрай складно. На поточний момент зазвичай в якості параметрів які вимірюються використовують різні людські риси: відбитки пальців, голос, райдужна оболонка очей, стиль натискань на клавіші, почерк і риси обличчя. Кожна характеристика надає змогу точно ідентифікувати людину з поміж мільйонів інших людей, бо немає жодної пари людей в якої співпали б перелічені вище біометричні дані. Саме тому якщо припустити що дана система вже існує, можна намалювати собі у свідомості щасливу картинку, в якій кожен

користувач має доступ лише до тієї інформації, до якої передбачений доступ системою. Це економило б мільярди доларів в економіці кожного року та давало в впевненість, що дані захищені. На даний час можна виокремити дуже багато спроб реалізувати дану задумку, проте на жаль дуже мало з них дають дійсно надійні результати. Даний дипломний дипломний проєкт дає змогу торкнутися цікавих сучасних інструментів в такій галузі науки, як наука про дані та використати такі славнозвісні методи, як глибокі нейронні мережі.

Актуальність теми: тенденція зростання кількості успішних кібер атак та значні фінсові збитки різних компаній світу, розсекречення військових та державних таємниць.

Мета розробки: створення веб застосування для ідентифікації користувача за голосом.

Завдання розробки:

– створення веб додатку, який надає можливість завантажити файл з голосом людини та пройти ідентифікацію;

– створення веб додатку, який надає можливість пройти ідентифікацію людини в реальному часі методом запису голосу;

– створення веб додатку, який надає можливість додавання файлів з голосом людини у створену базу данних;

– створення веб додатку, який надає можливість додавання користувача у базу данних на основі голосу записаного у реальному часі;

– створення та навчання нейронної мережі, яка буде ядром ідентифікації, яке можна буде інтегрувати у будь-яку систему;

– написання REST API на бекенді системи для можливості взаємодії з системою через інтерфейс API;

– створення веб додатку, який надає можливість додавання користувача у базу данних на основі голосу записаного у реальному часі;

Практичне значення одержаних результатів: розроблене веб застосування дозволить додати захисту до будь-якої електронної системи, де потрібна ідентифікація користувача. Більш того, ядро даної системи можна буде впровадити до будь-якого технологічного рішення – чи до веб додатку, чи до автономного домофону на КПП.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Успішність та надійність системи індентифікації користувача за голосом в першу чергу залежить від точності моделі, яка лежить в основі програми. Існує три етапи роботи з мовою: попередня обробка аудіофайлу, виокремлення фіч з обробленого файлу та безпосередньо верифікація. На кожному з етапів необхідно досягти найбільшої точності, яка можлива.

Тобто на попередній обробці необхідно отримати пайплайн в результаті якого прибираються шуми, похибки мікрофону та максимально зберігаються фічі аудіофайлу.

На другій стадії потрібно розробити алгоритм який дозволить виокремити фічі які максимально відрізняються одна від одної в аудіофайлах різних користувачів.

На третьому етапі необхідно отримати модель, яка матиме максмальну точність при тестуванні. 90% даних треба взяти для тренування та валідації, 10% даних для тесту. Для вирішення данної задачі використовувались різні підходи в поєднанні з різними технологіями. Робота над даним питанням розпочалась ще в 1937 році і продовжується донині.

1.2 Змістовний опис і аналіз предметної області

Найбільш надійними методами обмеження доступу користувачів є обмеження на основі біометричних даних. Використання систем розпізнавання користувача є найбільш економічним та природним способом вирішення питання несанкціонованого доступу до комп'ютера або до систем передачі

інформації, а також проблеми багаторівневого контролю доступу до інформаційних або мережевих ресурсів.

Зазвичай системи розпізнавання диктора поділяють на дві категорії: підтвердження особистість мовця (верифікація особистості) або визначення особистості із заданого, обмеженого списку людей (ідентифікація особистості). Ідентифікація та верифікація особистості за голосом є основними напрямками розвитку технології обробки мови на даний час.

Голос людини виникає при проходженні повітря з легенів через трахею в гортань, повз голосових зв'язок, далі в глотку, рот і носову порожнину. Коли хвиля звуку проходить через мовний тракт, її частотний спектр змінюється під дією коливань мовного тракту. Коливання мовного тракту називаються формантами. Системи верифікації та ідентифікації диктора зазвичай розпізнають відмітні ознаки мовного сигналу, які відображають індивідуальну особливість м'язової активності мовного тракту особистості. Розглянемо більш детально проблему ідентифікації користувача за голосом.

Робота системи проводиться у два етапи: реєстрація голосу мовця, тобто занесення його до бази данних користувачів, та безпосередньо ідентифікація. Під час цього процессу за допомогою різних методів з аудіофайлу витягується ряд особливостей його голосу, так званих фіч і заноситься до бази даних. На етапі ідентифікації отриманий зразок мови порівнюється з раніше створеними голосовими друками для визначення найкращої відповідності.

Весь процес обробки мовного сигналу можна розбити на кілька основних етапів:

- попередня обробка сигналу;
- виділення критеріїв (фіч);
- розпізнавання диктора.

Кожен етап представляє алгоритм або деяку сукупність алгоритмів, що в підсумку дає необхідний результат.

Головні риси голосу формуються трьома головними властивостями: механікою коливань голосових складок, анатомією мовного тракту і системою управління артикуляцією.

Головні ознаки, за якими приймається рішення про особистості диктора, формуються з урахуванням усіх чинників процесу речеобразования: голосового джерела, резонансних частот мовного тракту і їх затуханий, а також динамікою управління артикуляцією. Якщо розглянути джерела докладніше, то в якості голосового джерела входять: середня частота основного тону, контур і флуктуації частоти основного тону і форма імпульсу збудження. Спектральні характеристики мовного тракту описуються обвідної спектра і його середнім нахилом, формантного частотами, довготривалим спектром або кепстра. Крім того, розглядається також тривалість слів, ритм (розподіл наголосів), рівень сигналу, частота і тривалість пауз. Щоб визначити ці характеристики доводиться використовувати досить складні алгоритми, але так як, наприклад, похибка формантних частот досить велика, для спрощення використовуються коефіцієнти кепстра, які обчислюють за обвідної спектра або передавальна функція мовного тракту, знайдена методом лінійного передбачення. Крім згаданих коефіцієнтів кепстра також використовуються їх перші і другі різниці за часом. Цей метод був вперше запропонований в роботах Девіса і Мермельштейн.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Вперше роботи з розпізнавання мови з’явились ще у середині минулого століття. Перша система була створена на початку 1950-х років: розробники мали на меті розпізнавати цифри. Система могла розпізнавати цифри сказані одною людиною.

До кінця того дисятиліття з’явились системи, які здатні були розпізнавати голосні літери незалежно від спікера. А вже у 70-х роках почали використовувати нові методи, які дозволили отримати кращі результати – методом динамічного програмування та методом лінійного предбачення.

У 80-х роках наступним кроком у розвитку систем розпізнавання голосу стало використання прихованих марковский моделей (Hidden Markov Models - HMM). У цей час починають з'являтися перші великі програми з розпізнавання голосу, як наприклад, Kurzweil text-to-speech. В кінці 80-х також стали застосовуватися методи штучних нейронних мереж (Artificial Neural Network - ANN). У 1987 році на ринку з'явилися ляльки Worlds of Wonder's Julie doll, які були здатні розуміти голос. А ще через 10 років Dragon Systems випустила програму «NaturallySpeaking 1.0»

Нині створено багато бібліотек для розпізнавання голосу з відкритим кодом. Кожна з них відповідає за окремі частини повного пайплайну розпізнавання. Більшітсь з них написані на Python C, C++ та Java. Хоча деколи зустрічаються рішенні написані на застарілих мовах програмування типу Perl.

C – процедурна, універсальна, імперативна мова програмування розроблена ще у 1972 році Денісом Рітчі. Особливістю даної мови програмування є забезпечення низькорівневого доступу до оперативної пам’яті.

Хоча мова за свою структурою та поведінкою схожа на ассемблер, вона є машинно-незалежною та може легко компілюватися на величезній кількості операційних систем та апаратних платформ. Серед бібліотек, написаних на C та призначених для розпізнавання голосу можна виділити ббліотеку rnnoise призначений для подавлення стороннього шуму та libfvad призначену для виокремлення ділянок аудіофайлу де є голос та libmfcc призначену для виокремлення ознак з відрізків аудіофайлу.

C++ - мова програмування високого рівня, яка підтримує кілька парадигм програмування: процедурну, об'єкто-оріїнтовну та узагальнену. Створена була ще у 1983 році Б'ярном Страуструпом на основі мови С. Її використовують для написання драйверів, високопродуктивних клієнтських та серверних рішень.

Java – мова програмування високого рівня, яка була створена на основі об'єктної моделі C++ у 1995 році. Дана мова програмування також є платформо незалежною. Основна відмінність Java від C++ полягає у тому, що в ній усунута можливість виникнення деяких конфліктних ситуацій, що могли з'явитись через помилки розробника програмного забезпечення і спрощено процес розробки об'єктно орієнтовних програм.

Python - інтерпретована мова програмування високого рівня, яка має дуже велику кількість обгорток на готові рішення написані на інших мовах програмування (Cython, IPython та ін.) та власні бібліотеки з відкритим кодом, які широко використовуються у рішеннях написаних власне на Python. Найбільш популярні та необхідні бібліотеки які використовуються у рішеннях розпізнавання за голосом:

– numpy, який прозначиней для роботи з багатовимірними масивами та використання різних математичних функцій;

– sciру, призначений для вирішення задач оптимізації, розв'язання диференціальних рівнянь, обробки зображень та ін.;

– seaborn, matplotlib, plotly для візуалізації даних;

– pandas для представлення даних у вигляді таблиць, маніпуляцій з ними, чистки даних, їх трансформації;

– tensorflow, keras, sklearn, scikit-learn для побудови складних моделей та нейронних мереж.

Серед найбільш відомих допоміжних бібліотек написаних на Python які використовуються у розпізнаванні за голосом можна виокремити librosa.

Якщо мова йде про написання веб серверу мовою Python, то найбільш класичними технологіями для вирішення цієї задачі можна вважати веб-фреймворки Flask, Django та Tornado.

Flask являється найпростішим з названих вище фреймворків. Він не вимагає спеціальних засобів чи бібліотек а також не має рівня абстракції для роботи з базами даних.

Якщо мова йде про вибір СКБД для проекту, то можна+ виокремити найбільш вживані: MySql, PostgreSQL, MsSql, Oracle

Даний дипломний проект був написаний на мові Python, з використанням бібліотек librosa, tensorflow, keras, різноманітних обгорток на бібліотеки C. Веб сервер написаний за допомогою фреймворку Flask, а за СКБД обрано MySql.

1.3.2 Аналіз відомих програмних продуктів

Найуспішнішим та єдиним занайденим рішенням даної проблеми на даний час є продукт Speaker Identification API від компанії Microsoft. Даний продукт інтегровано в платформу Microsoft Azure. Найголовнішою перевагою використання даного продукту є простота у використанні – надається можливість використання по API, швидкий старт та можливість користуватися сервісами 24/7. Головною складністю використання є його ціна.

Всі інші знайдені рішення були або лише у вигляді статтей або у вигляді недопрацьованого коду на GitHub.

Тим не менше існує декілька доволі успішних проектів на споріднену з даною проблемою темою. Перш за все це нейронна мережа від компанії Google UIS-RNN. Розробники данного продукту працювали над проблемою діарізації дикторів. З данною нейронною мережею була досягнута точність 85% на великому та різношаровому датасеті. В основі цього рішення лежить нейронна мережа, яка побудована на принципі навчання з учителем.

Іншим не менш відомим продуктом у вузьких колах є Speaker Recognition API від компанії Google, який досяг точності 97% в питанні транскрибації людської мови у текст. Серед явних переваг використання даного продукту – простота та швидкий старт. Серед труднощів у використанні – платна підписка.

Доволі цікавим продуктов який уже існує доволі давно є написаний на C++ проект Kaldi, який також надає ядро та інтерфейс для розпізнання тексту диктора. Головною його перевагою є відкритий код, можливість написання власних бібліотек та будь-якої модифікації під власний продукт. Kaldi включає в себе велику кількість інструментів для обробки людського голосу. Приємним фактом є те, що існує написана на Python високорівнева обгортка до цього проекту.

1.4 Аналіз вимог до програмного забезпечення

Система повинна містити два типи користувачів – адміністратор та користувач.

Адміністратор – це користувач з необмеженими правами, який може виконувати багато функцій: очищення бази данних користувачів, видалення конкретного користувача, додавання користувача, створення нової таблиці для певної групи людей та організацій.

Користувач має доступ до інтерфейсу веб додатку в якому реалізовані функції ідентифікації користувача з завантаженого файлу або з запису голосу через мікрофон у реальному часі, додавання його в базу даних на основі завантаженого файлу або записаного в реальному часі.

В системі повинні бути реалізовані наступні функції:

– інтерфейс для двох сценаріїв використання веб-додатку: через файл та в режимі запису голосу в реальному часі;

– виведення ідентифікованого користувача на екран та процент впевненості системи в правильній відповіді.

1.4.1 Розроблення функціональних вимог

Структурна схема варіантів використання наведена у графічному матеріалі.

В системі передбачено наступні варіанти використання, які описані у таблицях 1.1 – 1.14:

Таблиця 1.1 - Варіант використання UC001

Назва	Завантаження данних
Опис	Користувач повинен мати змогу завантажувати аудіофайл у систему
Учасники	Користувач.

Таблиця 1.2 - Варіант використання UC002

Назва	Ідентифікація користувача
Опис	Користувач може проходити ідентифікацію
Учасники	Користувач.

Продовження таблиці 1.2

Передумови	Створена база даних для користувачів, зроблене базове заповнення
Постумови	Отримано ідентифіковану особу та відсоток впевеності системи
Основний сценарій	Користувач натискає кнопку запису голосу. По завершенню запису з'являється кнопка «Ідентифікувати». Після натискання цієї кнопки виконується ідентифікація

Таблиця 1.3 - Варіант використання UC003

Назва	Додавання користувача в базу даних
Опис	Користувач може додавати себе в базу даних
Учасники	Користувач.
Передумови	Створена база даних для користувачів, зроблене базове заповнення
Постумови	Користувача додано в базу даних
Основний сценарій	Користувач натискає кнопку запису голосу. По завершенню запису з'являється кнопка «Додати в базу даних». Після натискання цієї кнопки виконується додавання в базу даних

Таблиця 1.4 - Варіант використання UC004

Назва	Вибір сценарію ідентифікації

Продовження таблиці 1.4

Опис	Користувач може обирати один із двох сценаріїв ідентифікації – за допомогою завантаженого файла або запису голосу в реальному часі
Учасники	Користувач
Передумови	Завантажена основна сторінка веб додатку
Постумови	З'являється можливість додавання в базу даних або верифікації
Основний сценарій	Користувач обирає завантаження файлу або запис голосу в реальному часі

Таблиця 1.5 - Варіант використання UC005

Назва	Створення бази даних.
Опис	Адміністратор може створити нову таблицю користувачів
Учасники	Адміністратор.
Передумови	З’явлася необхідність занесення до системи нової групи людей.
Постумови	В базі даних створено нова таблиця для нової групи користувачів.
Основний сценарій	Адміністратор під'єднався до бази даних через будь-який зручний клієнт, ввів команду створення таблиці

Таблиця 1.6 - Варіант використання UC006

Назва	Видалення користувача

Продовження таблиці 1.6

Опис	Адміністратор може видяляти користувачів з таблиці
Учасники	Адміністратор.
Передумови	З'явилася необхідність видалення користувача з таблиці бази даних
Постумови	Користувач видалений з таблиці
Основний сценарій	Адміністратор під'єднався до бази даних через будь-який зручний клієнт, ввів команду видалення користувача

Таблиця 1.7 - Варіант використання UC007

Назва	Додавання користувача
Опис	Адміністратор може додавати користувачів у базу даних
Учасники	Адміністратор.
Передумови	З'явилась необхідність додання користувача в базу даних.
Постумови	Користувач доданий у базу даних
Основний сценарій	Адміністратор під'єднався до бази даних через будь-який зручний клієнт, ввів команду додавання користувача в базу даних

Функціональні вимоги додатку описано наступними таблицями.

Таблиця 1.8 - Опис функціональної вимоги REQ001

Номер	REQ001

Продовження таблиці 1.8

Назва	Обрання способу передачі даних
Опис	Користувач може обирати варіанти передачі голосу – завантаженням через файл або записом в реальному часі

Таблиця 1.9 - Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Ідентифікація за файлом
Опис	Користувач має змогу ідентифікувати особистість через завантажений файл з голосом

Таблиця 1.10 - Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Запис голосу в реальному часі
Опис	Користувач має змогу записати свій голос в реальному часі через веб інтефес додатку

Таблиця 1.11 - Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Ідентифікація за записаним голосом
Опис	Користувач може проходити процедурур ідентифікації за записаним голосом

Таблиця 1.12 - Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Додавання в базу даних на основі завантаженого файлу
Опис	Користувач може додавати себе в базу даних на основі завантаженого раніше файлу

Таблиця 1.13 - Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Додавання в базу даних на основі записаного голосу
Опис	Користувач може додавати себе в базу даних на основі записаного раніше голосу

Таблиця 1.14 - Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Підключення адміністратору до бази данних, можливість введення команд
Опис	При підключенні адміністратору до бази данних він має можливість створювати таблиці груп користувачів, додавати та видаляти користувачів

Взаємозв’язки між вимогами клієнтського застосунку відображені на рисунку 1.1.

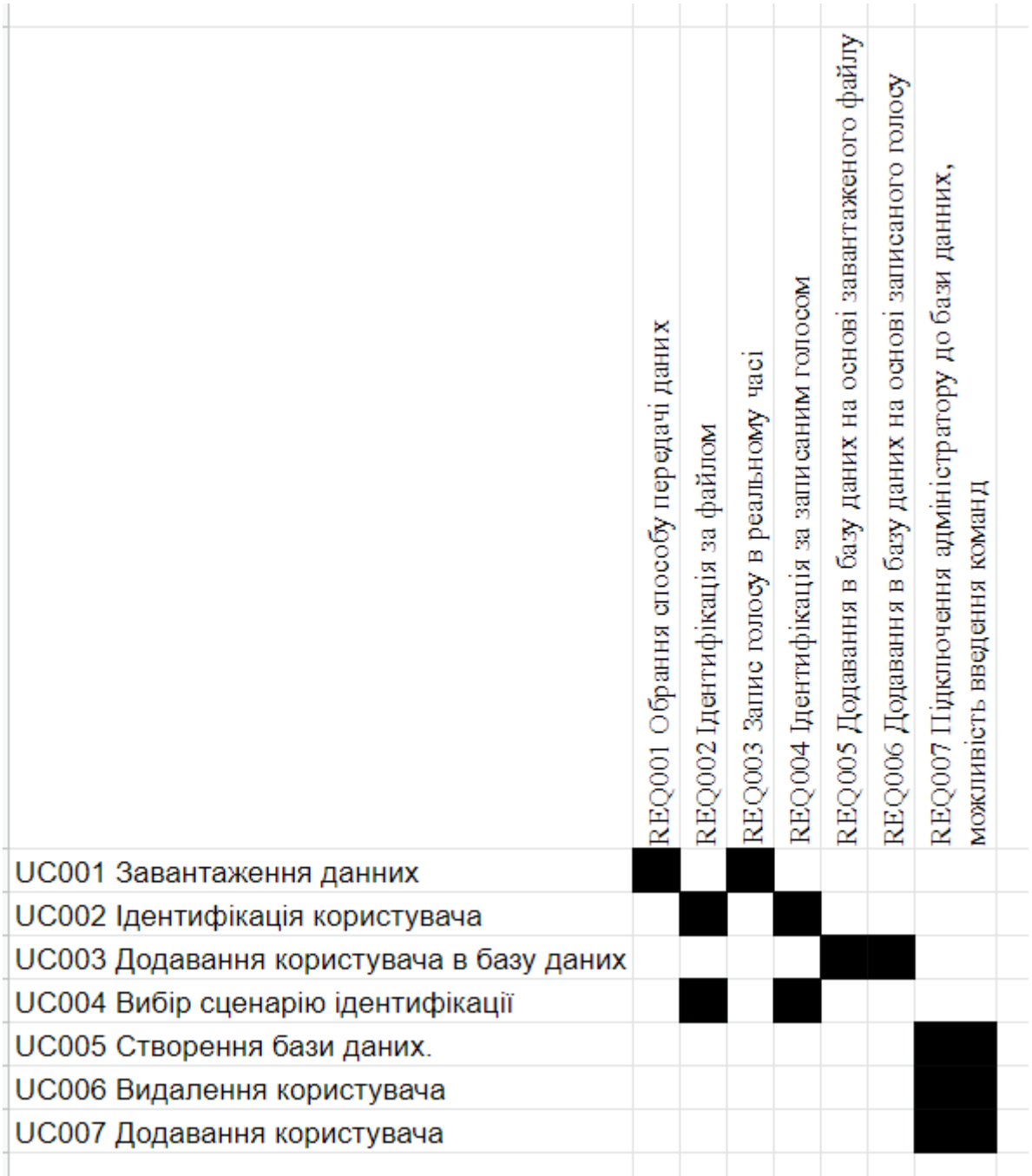

Рисунок 1.1 - Матриця залежності між вимогами застосунку і варіантами використання

1.4.2 Розроблення нефункціональних вимог

Вимоги до інтерфейсу:

– клієнтська частина програмного забезпечення має бути доступною в будь-якому браузері;

– необхідна можливість відображення результату тестів на інтерфейсі.

Апаратні та програмні вимоги:

– застусування повинно запускатись на машині з відеокартою Nvidia;

– застосування має працювати з операційними системами Ubuntu 16.04, Ubuntu 18.04.

Операційні вимоги:

– відновлюваність:

1) в разі збою необхідна можливість повторного виконання необхідної дії після перезавантаження сервісу;

2) в разі пошкодження бази даних необхідна можливість відтворення данних з бекапу;

3) в разі пошкодження бази даних необхідна можливість відтворення данних з бекапу;

– збереження даних:

1) система повинна зберігати голосові зліпки всіх користувачів до моменту видалення їх з бази данних Адміністратором;

– продуктивність:

1) кількість запитів в сервері не перевищує 1 на 5 секунд (при базі данних з 450 користувачами);

2) час відповіді системи на запит ідентифікації не перевищує 6 секунд;

3) час відповіді системи на команду занесення користувача в базу даних не перевищує 10 секунд.

1.4.3 Постановка комплексу завдань модулю

Розрблюване програмне забезпечення призначене вирішувати задачу обмеження доступу для різних організацій та установ шляхом голосової ідентифікації.

Мета створення даної роботи – надати фінансовим, юридичним, військовим та іншим установам сервісу для захисту від несанкціонованого доступу до інформації.

Для досягнення мети даної роботи система повинна вирішувати наступні задачі:

– створення різних таблиць для різних підрозділів конкретної установи;

– можливість створення повної бази даних на основі записів голосу користувачів;

– можливість додання нових користувачів у базу даних.

Сервіс повинен працювати на пристроях зі встановленою операційною системою Ubuntu 16.04 і вище, а також на пристроях, де встановлені будь-які інші операційні системи Unix з можливими незначними порушеннями.

1.5 Математичне забезпечення

В основі даної роботи лежить чималий бекграунд з математики. Почнемо з фундаментальної операції – дискретне перетворення Фур’є[7]. Це процедура, яка використовується для обробки дискретних сигналів. Зокрема, за її допомогою визначають гармонічний або частотний склад дискретних сигалів. Проте в його основі лежить неперервне перетворення Фур’є, яке визначається формулою:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft}\, dt$$

, де

- X(m) – m-ий компонент ДПФ
- n – часовий індекс вхідних відліків
- m – індекс ДПФ в частотній області
- N – кількість відліків вхідної послідовності
- x(n) – послідовність вхідних відліків

Властивості ДПФ:

– пердіодичність (дискретний спктр періодичний, а один період спектра містить N гармонік);

– симетричність (амплітудний спектр дійсного сигналу з парною кількістю відліків є сполучно-симетричним відносно k, а частота гармоніки рівна половині частоти дискретизіції. Це відповідає критерію Найквіста;

– модульність;

– лінійність (ДПФ заелементної суми послідовностей однакової довжини N дорівнює сумі перетворень окремих послідовностей;

– зсув (звув послідовності на деяку кількість тактів рівнозначний множенню або діленню спектра вихідної послідовності на експоненту);

– додавання нулів (при практичній реалізації припустиме доповнення сигналу нульовими відрізками, якщо довжина менше розміру ДПФ);

– ДПФ добутку послідовностей (спектр послідовності, яка отримана в результаті поелементного множення двох послідовностей рівної довжини, формується вираженням, що являє собою циклічну згортку спектрів вихідних послідовностей;

– енергія спектра (спектр ДПФ зберігає інформацію про енергію вихідного сигналу, що відтворено в дискретному аналогу рівності Парсеваля.

Не менш важливою темою для цього розділу є мел кепстральні частотні коефіцієнти[3], так як це одна з фіч, яка використовується в даному проєкті. Отже, мел кепстральні частотні коефіцієнти - представлення короткочасного спектру потужності звуку, заснованого на лінійному перетворенні косинусу спектру потужності журналу в нелінійній шкалі мель. Іншими словами, якщо у нас є спектрограма голосу людини, ми можемо виокремити з неї окремі фічі, такі як потужність звуку, його частота, для подальшого пайплайну моделі.

Значення MFCC отримуються з модуля спектру (модуль значень швидкого перетворення Фур'є вхідного сигналу) із застосуванням банку фільтрів, які рівномірно розподілені на «викривленій», за відповідним законом, частотній шкалі. Далі отриманий спектр зважується банками фільтрів, отримується набір значень, який логарифмується і потім декорелюється дискретним косинусним перетворенням (ДКП). Результатом останньої дії є вектор кепстральних коефіцієнтів. Описаний процес продемонстровано на рис. 1.2

Рисунок 1.2 - Алгоритм отримання кепстральних коефіцієнтів

Якщо для деформації частотної шкали використовується Мел-шкала, то отримані коефіцієнти називають Мел-кепстральними (MFCC), якщо Барк-шкала – Барк-кепстральними (BFCC), якщо частотна шкала не деформується – отримуємо UFCC (Uniform-frequency Cepstral Coefficients). Мел-шкала, це емпірична шкала, що ґрунтується на людському відчутті частоти звуку, була запропонована Стівенсом і Волкманом в 1937 році. Шкала була отримана в

результаті експериментів, в яких, випробуваних просили скорегувати сигнал , який вони чують таким чином, щоб його висота стала в два рази нижчою. В результаті була отримана шкала, в якій 1000 Мел відповідає «висоті» звуку з частотою 1 кГц і подвоєння Ме створює відчуття сприйняття подвоєння висоти звуку.

Декілька десятиліть тому з’явилось поняття глибоких нейронних мереж. Раніше вивчення даної теми обмежувалось обчислювальною потужністю комп’ютерів. Максимум, на який були здатні комп’ютери – 7 слоїв у нейронці. Поступово обчислювальна потужність дала змогу глибше зануритись науковцям з усього світу в цю тему. Одним з різновидів нейронних мереж є клас рекурентних нейроних мереж. Вихід першої клітини нейронної мережі являється входом для наступної. Підвидом рекурентних нейронних мереж є клас нейронних мереж LSTM[4], що в перекладі з англійської означає Довга короткочасна пам'ять. Це особлива архітектура рекурентних нейронних мереж (RNN), яка була запропонована Зеппом Хохрайтером та Юргеном Шмідгубером ще у 1997 році. До речі ще у 2003 році мережі ДКЧП досягли найкращого результату у проблемі автоматичного розпізнавання мовлення, і були головною складовою мережі яка досягла рекордні 82,3% точності.

Традиційна ДКЧП:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Для мінімізації загальної похибки ДКЧП на послідовностях для тренування може застосовуватися ітеративний градієнтний спуск, що використовує

зворотне поширення в часі, для зміни кожного вагового коефіцієнту пропорційно до його похідної по відношенню до похибки.

Наступною темою про яку варто детально поговорити в контексті даної дипломної роботи – Виявлення голосової активності (VAD). Це процес виявлення голосової активності у вхідному акустичному сигналі для відділення активного мовлення від фонового шуму або тиші. Голос, інтерпретований як шум, може породжувати «вирізки» з розмови (chipping). Фон, що інтерпретується як голос, призводить до зниження ефективності компресії. Використання механізму VAD[5] (або Silence Suppression) дозволяє економити на передавання даних у мережі зв'язку, так як перерва у мовленні (визначається за рівнем сигналу) НЕ оцифровується і не кодується і таким чином «порожні» пакети з тишею не передаються по мережі.

Загалом існує декілька класичних підходів VAD. Перший метод працює на основі нульової швидкості перетину та енергії. Він - простий і швидкий метод підходу для поділу даної мови сигнали на голосові та беззвучні класи. Метод працює на комбінації. нульової швидкості перетину та розрахунків енергії. Нульова швидкість схрещування може бути визначена як кількість разів послідовних проб у мовному сигналі мають різні алгебраїчні знаки або амплітуду сигналу перетинає значення нуля. Нульова швидкість схрещування вказує на наявність або відсутність мови на вході сигнал. Якщо нульова швидкість перетину висока, кадр вважається беззвучним і якщо він низький, кадр вважається озвученим кадром.

Другий метод заснований на короткочасній енергії. Короткочасний підрахунок енергії - ще один параметр, який використовується в класифікації озвучені та неозвучені сегменти. Якщо енергія вхідного кадру висока, то кадр класифікується на озвучений кадр і, якщо енергія вхідного кадру низький, він класифікується на необмежений кадр. У цьому способі використовується вікно,

що забиває, яке дає значне загасання зовні смуга проходу порівняно з прямокутним вікном.

Ще одним потужним методом вважається «Світлодіод: лінійний VAD на основі енергії». У попередньому методі поріг залишався постійним через весь процес. Цей метод працює за принципом оновлення порогового значення адаптивно.

Модифікацією цього методу можна вважати ALED[6]: Адаптивний лінійний детектор на основі енергії. Цей метод є вдосконаленням попереднього методу лінійної енергетики детектор (світлодіод). Коефіцієнт 'p' обмежений постійним

значенням, нечутливий до різної статистики шуму. Щоб подолати це обмеження, Er, поріг енергії обчислюється за допомогою другого порядку статистики необроблених кадрів.

Наступним методом, який варто висвітлити в даній курсовій роботі є VAD на основі статистичних метрик. Цей метод описує статистичний метод, який використовує сигнал-шум міра відношення для виявлення мовного сегмента у вхідному сигналі. Метод включає оцінку низькодисперсного спектру та адаптивного порогового механізму виявлення озвучених сегментів у вхідному сигналі. Очікувана спектральна щільність потужності шуму та дисперсія «відношення сигнал до шуму» міра оцінюється з мовленнєвих періодів. Адаптив обчислення порогових значень покращує продуктивність VAD.

1.6 Висновки до розділу

В розділі 1 проведено:

– аналіз існуючих технічних засобів і відомих рішень;

– аналіз відомих програмних продуктів, їх переваг та недоліків;

– аналіз вимог до програмного забезпечення – розроблені функціональні та нефункціональні вимоги.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Даний веб сервіс повинен мати клієнт серверну архітектуру. Розроблений для операційної системи Ubuntu 16.04 і вище.

Ubuntu – зазвичай перша операційна система кожного нового користувача Unix. Вона має базовий мінімальний необхідний набір програм та графічну оболонку.

Це програмне забезпечення добре працює і на інших дистрибутивах Unix.

В основі архітектури лежить клієнт серверна схема взаємодії користувача та ядра системи. Серверна частина додатку написана мовою програмування Python на основі фреймворку Flask. Якщо говорити про Flask, то перш за все треба виділити простоту, гнучкість у роботі. Він дозволяє програмісту самостійно обирати як саме реалізовані ті чи інші речі.

За СКБД було прийнято рішення взяти MySql. Дана система характеризується стійкістю, простотою використання та значною швидкістю доступу до даних.

Ядром програми можна вважати модель побудовану на основі навченої нейронної мережі для ідентифікації користувача за голосом. Про неї поговоримо більш детально.

Для успішної ідентифікації нейронною мережею користувача необхідно провести ряд перетворень над вхідним файлом для виокремлення ознак.

По перше на якість цих ознак впливають зовнішні шуми. При ідентифікації в офісі, на вулиці, у будь-якому іншому місці до аудіофайлу

можуть потрапити сторонні звуки, такі як натискання клавіш, гудіння кавоварки або чхання колеги, який нещодавно прилетів із Китаю чи Італії. Тому першим кроком буде пропускання нашого аудіофайлу через систему для усунення шуму. Робота над даним напрямком була розпочата ще дуже давно. За цей час виокремлись основні етапи класичного підходу. Їх можна виокремити в рисунок 2.1

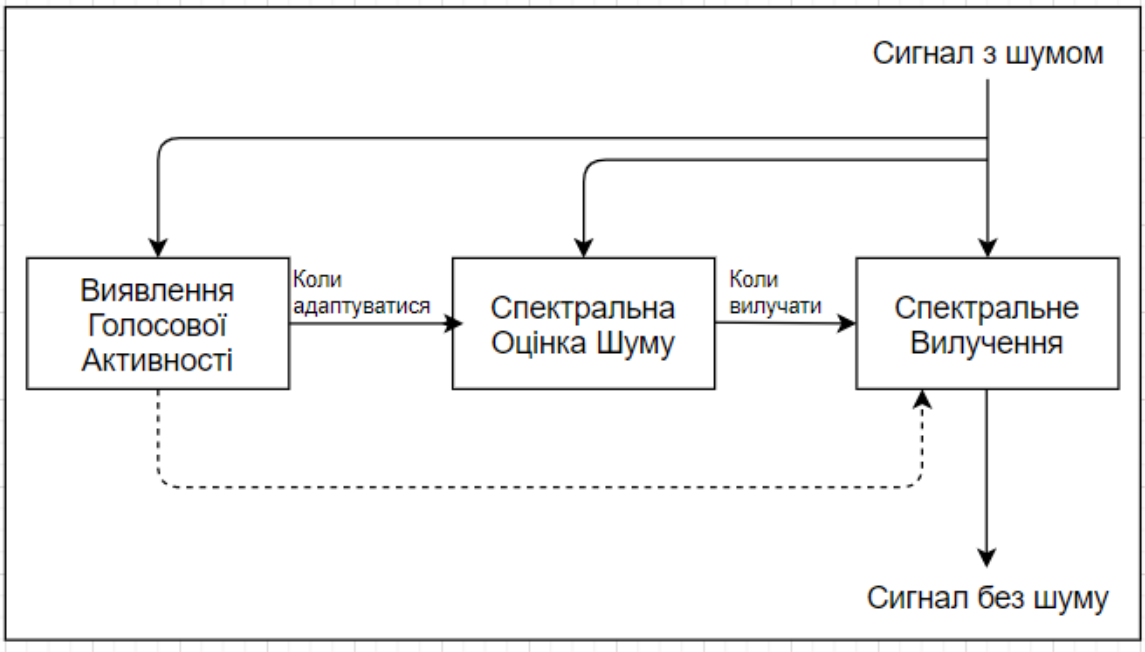

Рисунок 2.1 – Високорівневий алгоритм усунення шуму в аудіозапису

Для вирішення цієї проблеми будемо використовувати бібліотеку rnnoise, написану на C. В основі її використання гібридний підхід, який задіює одночасно і добре відомі методи шумозаглушення, і глибинне навчання для заміни тих компонентів, які важко настроюються в звичайних системах. Блок схема її роботи наведена на рисунку 2.2.

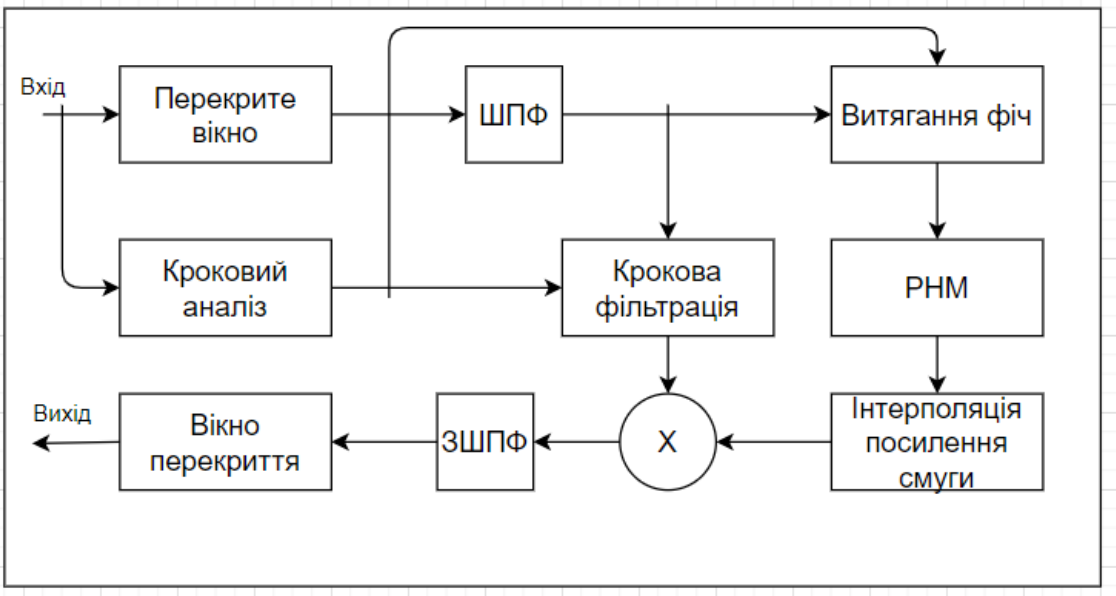

Рисунок 2.2 – Схема роботи шумозаглушної системи rnnoise

На відміну від шумозаглушної системи компанії Google, дана система не використовує значних потужностей системи та може запускатися майже на будь-якому суперкомп'ютері, а її якість нічим не поступається.

Отже, з проблемою шумів розібрались. Уявіть тепер, що деякі користувачі мають голосний, а деякі тихий голос. Спектрограма їх аудіофайлів буде сильно від цього залежати. Щоб це не впливало на якість кінцевої моделі використаємо звичайну нормалізацію векторів. Даний підхід компенсує різницю між голосним звучайнням та тихим.

Після видалення шуму та нормалізації векторів наступним питанням постає на яких ділянках аудіофайлу є голос, а на яких немає. Зазвичай в цьому пункті використовують підхід Виявлення Голосової Активності.

Після виявлення ділянок з голосом наступним пунктом у нашому пайплайні є Віконе перетворення Фур’є

Віконне перетворення Фур’є – це перетворення Фур'є, яке застосовується для визначення синусоїдальної частоти та фазового вмісту локальних ділянок сигналу, оскільки він змінюється з часом. На практиці дана процедура полягає в

тому, щоб розділити більш тривалий часовий сигнал на більш короткі сегменти однакової довжини, а потім обчислити перетворення Фур'є окремо на кожному коротшому сегменті. Це розкриває спектр Фур'є на кожному коротшому сегменті.

Далі маємо сигнал готовий до подачі на вхід самої нейронної мережі. В основі розробленої нейронної мережі є так звана LSTM – Довга короткочасна пам’ять. На відміну від традиційних нейронних мереж дана архітектура чудово справляється з задачею передбачення часових рядів та класифікації. Навчання даної нейронної мережі відбувалося на зібраному датасеті з ресурсу Яндекс. Толока. Було зібрано приблизно 450 зразків людської мови.

На виході нормалізовуємо вектор фіч та маємо готовий зліпок голосу користувача, який можна зберігати в базу данних або ідентифікувати користувача.

2.2 Архітектура програмного забезпечення

Діаграма класів програмного продукту наведена у графічному матеріалі. Детальний опис класів та функцій наведений нижче в таблицях.

Таблиця 2.1 - Опис класів сервісу

Клас	Опис
SpeechEmbeder	Клас спроектованої нейронної мережі, яка відповідає за ідентифікацію
Hparam	Клас, який відповідає за взяття інформації з гіперпараметрів
Dotdict	Клас, який є основою класу Hparam

Продовження таблиці 2.1

Frame	Клас, який реалізує створення фреймів на етапі Виявлення голосової активності.

Таблиця 2.2 - Опис методів класів сервісу

Клас	Метод	Опис
SpeechEmbeder	__init__	Конструктор. Ініціалізація початкових значень атрибутів. Параметри: - nmels – розмірність мел-частотних коефіцієнтів спектруму - num_layers – кількість прихованих слоїв LSTM
SpeechEmbeder	forward	метод навчання нейронної мережі, який відповідає за проходження однієї епохи
Dotdict	__init__	Конструктор. Ініціалізація початкових значень атрибутів. Параметри: - dct – словник початкових значень
Hparam	__init__	Конструктор. Ініціалізація початкових значень атрибутів. Параметри: - file – конфігураційний файл заданої системи

Продовження таблиці 2.2

Frame	__init__	Конструктор. Ініціалізація початкових значень атрибутів. Параметри: - bytes – аудіофайл - timestamp – заданий проміжок часу - duration – задана тривалість фрейму

Таблиця 2.3 - Опис статичних атрибутів класів застосунку

Клас	Атрибут	опис
SpeechEmbeder	LSTM_stack	Атрибут, що відповідає за шар нейронної мережі
SpeechEmbeder	projection	Атрибут, що зберігає в собі проекцію свого класу
Frame	bytes	Атрибут, що заберігає в собі безпосердньо файл з аудіо
Frame	timestamp	Атрибут, що заберігає в собі безпосердньо файл з аудіо
Frame	duration	Атрибут, що заберігає в собі задана тривалість фрейму

Таблиця 2.4 - Опис функцій модулів застосунку

Модуль	Функція	Опис
app	allowed_file	Перевіряє чи є завантажений файл голосовим. Параметри: - ім’я файлу

Продовження таблиці 2.4

app	upload_form	Відображення сторінки веб-додатку
app	upload_file	Відображення результатів проведеного експерименту ідентифікації Параметри: - використані методи: GET, POST та ін.
app	not_found	Опрацювання помилки 404. Повернення відповідного інформаційного json-у Параметри: - помлка
create_queries_table	make_queries	Створення таблиці з історіїю запитів до бази даних
db_export_to_mysql	add_to_db	Функція, що створює базу даних користувачів на основі векторів ембедінгів. Параметри: - db_root – параметр, що відповідає за місце розташування списку ембедінгів
hparam	load_hparam	Функція, що завантажує параметри з конфігураційного файлу у програму Параметри: - відносний шлях розташуваня файлу

Продовження таблиці 2.4

identify	get_STFTs	Функція, що перетворює масив сегментів сигналу у вікна STFT довжиною 240мс з 50% перекриттям Параметри: - segs – сегменти аудіофайлу
identify	ffmpeg_read_pcm16	Функція, перетворює заданий аудіофал у масив чисел Параметри: - file – відносний шлях до файлу - sr – частота файлу.
identify	ffmpeg_resample_pcm16	Функція, що змінює вектор отриманого аудіо при зміні частоти Параметри: - audio – вектор аудіофайлу - sr_old – стара частота файлу - sr_new – нова частота файлуч
identify	ffmpeg_normalize	Функція, що нормалізує заданий аудіофайл з заданою частотою Параметри: - audio – вектор аудіофайлу - sr – частота файлу

Продовження таблиці 2.4

identify	append_centroid_to_db	Додавання центроїду в БД. Параметри: - cenroid – зліпок голосу користувача, вектор чисел - name – ім’я користувача - db_table – табличка бази даних, до якої слід додати користувача
identify	append_user_to_db	Додавання користувача в БД. Параметри: - cenroid – зліпок голосу користувача, вектор чисел - name – ім’я користувача - db_table – табличка бази даних, до якої слід додати користувача
identify	centroid_from_files	Отримання зліпку голосу(центроїду) користувача з заданого голосового файлу Параметри: - files – перелік файлів
identify	speaker	Ідентифіація користувача, отримання ім’я та відсотку впевненості Параметри: - verification_embeding – зліпок голосу користувача, вектор чисел - db_table – ім’я таблиці, в якій проводити ідентифікацію

Продовження таблиці 2.4

identify	identify	Те ж, що і функція speaker, але для консольного виводу
make_db_from_wav	db_from_wav	Створення таблиці з заданим ім’ям в базі даних, наповнення її користувачами Параметри: - mysql_db – ім’я таблички - шлях до аудіофайлів
test_db	accuracy	Тест отриманної точності моделі
VAD_segments	frame_generator	Генерація аудіо фреймів з PCM аудіо Параметри: - frame_duration_ms - довжина фрейму - audio – вектор аудіофайлу - sample_rate – частота файлу
VAD_segments	vad_collector	Фільтрує фрейми в яких немає голосу. Параметри: - sample_rate – частота файлу - frame_duration_ms - довжина фрейму - padding duration_ms – довжина підряд йдучих фреймів
VAD_segments	VAD_chunk_from_bytes	Створення відрізків з фреймів на основі бібліотеки виявлення голосової діяльності

Таблиця 2.5 - Опис таблиць бази даних

Таблиця	Опис
Speakers	У таблиці зберігаються зліпки голосів користувачів та інформація про них.

Таблиця 2.6 – Опис полів бази даних

Таблиця	Поле	Опис
Speakers	id	Первинний ключ таблиці Speakers. Тип: integer.
Speakers	Name	Ім’я користувача. Тип: varchar.
Speakers	Telephone	Телефон користувача. Тип: varchar.
Speakers	Centroid	Центроїд користувача. Тип: varbinary.

2.3 Конструювання програмного забезпечення

Написаний продукт має декілька сценаріїв використання для кінцевого користувача

Загальний алгоритм роботи з додатком:

– обрати шлях ідентифікації – через файл або через запис голосу в реальному часі. На головній сторінці веб додатку можна обрати ліву або праву чатину екрану;

– у вікні тестування за файлом натиснути кнопку «Завантажити файл»;

– після завантаження файлу з’являться дві кнопки – ідентифікація та додання користувача у базу даних При натисканні ідентифікація через декілька секунд буде виведено на екран ідентифікований користувач та ступінь впевненості системи у своєму рішенні. При натисканні додати у БД буде виведено повідомлення про успішне додання користувача у БД;

– при виборі запису голосу буде можливість обрати одну з двох кнопок – «Ідентифікація» та «Додавання у БД». При натисканні ідентифікація з’виться вікно запису через мікрофон та таймер 10 сек. Після 10-ти секунд файл автоматично буде надісланий на сервер та ідентифікований. При натисканні додавання в БД з’виться вікно запису через мікрофон та таймер 30 сек. Після 10-ти секунд файл автоматично буде надісланий на сервер та доданий в БД.

2.4 Аналіз безпеки даних

База даних користувачів зберігатиметься у форматі векторів зліпків голосу користувачів. Це автоматично слугує надійним захистом приватних даних у разі компрометування бази даних.

Проте слабким місцем даної системи на даний час є передача аудіофайлів від клієнта до сервера. Зараз цей механізм реалізований через звичайні GET та PUT запити, які не захищені протоколом шифрування даних на кшталт HTTPS або TLS.

2.5 Висновки до розділу

В розділі 2 виконано моделювання та аналіз розробленого продукту, описано його архітектуру (специфікацію методів класів програмного забезпечення та функцій модулів) в діаграмі класів.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Одним з найважливіших етапів створення програмного забезпечення є його тестування. Існує декілька критеріїв поділу тестування на види. Наприклад, якщо хочемо поділити на типи в залежності від переслідуваних цілей, можна виокремити наступні види:

– функціональне тестування;
– нефункціональне тестування;
– те, що пов’язане зі змінами.

За ступенем ізольованості тестування можна поділити на:

– модуальне (компонентне) тестування;
– системне тестування;
– інтеграційне.

Розглянемо більш детально деякі види. Функціональне тестування фактично імітує використання системи. Воно передбачає аналіз функціональних характеристик та перевіряє відповідність між очікуваною та реальною поведінкою.

Нефункціоналне тестування має на увазі тестування нефункціональних вимог програмного забезпечення. Серед них можна виділити надійність, продуктивність, безпека, масштабованість тощо.

Отже, перед розгортанням рішення у робочому середовищі потрібно провести перелічені види тестів, створити звіт який включатиме в себе список та опис проведених тест-кейсів і прийняти рішення – розгортати додаток чи відправити на доопрацювання.

3.2 Опис процесів тестування

Список проведених тест кейсів та одержаних результатів наведено у таблиці 3.1

Таблиця 3.1 – Список тест кейсів та отриманих результатів

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
CF-1	Користувач може завантажити файл з коректним розширенням.	Користувач може завантажити файл з коректним розширенням.	Пройдено
CF-2	Користувач не може завантажити файл з некоректним розширенням.	Користувач не може завантажити файл з некоректним розширенням.	Пройдено
CF-3	Користувач може пройти тест ідентифікації на основі завантаженого коректного файлу натиснувши кнопку «Ідентифікувати»	Користувач може пройти тест ідентифікації на основі завантаженого коректного файлу натиснувши кнопку «Ідентифікувати»	Пройдено

Продовження таблиці 3.1

CF-4	Користувач може додати себе в базу даних на основі завантаженого коректного файлу ввівши своє ім’я та натиснувши кнопку «Додати в базу даних »	Користувач може додати себе в базу даних на основі завантаженого коректного файлу ввівши своє ім’я та натиснувши кнопку «Додати в базу даних »	Пройдено
CF-5	Користувач не може додати себе в базу даних на основі завантаженого коректного файлу не ввівши своє ім’я та натиснувши кнопку «Додати в базу даних »	Користувач не може додати себе в базу даних на основі завантаженого коректного файлу не ввівши своє ім’я та натиснувши кнопку «Додати в базу даних »	Пройдено

Продовження таблиці 3.1

CF-6	Наступні 10 секунд після натискання Користувачем кнопки «Запис для тесту» буде йти запис його голосу про що свідчиме коливання лінії	Наступні 10 секунд після натискання Користувачем кнопки «Запис для тесту» буде йти запис його голосу про що свідчиме коливання лінії	Пройдено
CF-7	Наступні 30 секунд після натискання Користувачем кнопки «Запис для бази даних» буде йти запис його голосу про що свідчиме коливання лінії	Наступні 30 секунд після натискання Користувачем кнопки «Запис для бази даних» буде йти запис його голосу про що свідчиме коливання лінії	Пройдено

Продовження таблиці 3.1

CF-8	Після 10 секунд запису голосу Користувача стане активною кнопка «Ідентифікувати»	Після 10 секунд запису голосу Користувача стане активною кнопка «Ідентифікувати»	Пройдено
CF-9	Після 10 секунд запису голосу Користувача не стане активною кнопка «Додати в базу даних»	Після 10 секунд запису голосу Користувача не стане активною кнопка «Додати в базу даних»	Пройдено
CF-10	Після 30 секунд запису голосу Користувача стане активною кнопка «Додати в базу даних»	Після 30 секунд запису голосу Користувача стане активною кнопка «Додати в базу даних»	Пройдено

Продовження таблиці 3.1

CF-11	Після 30 секунд запису голосу Користувача не стане активною кнопка «Ідентифікувати»	Після 30 секунд запису голосу Користувача не стане активною кнопка «Ідентифікувати»	Пройдено
CF-12	Якщо при додаванні в базу даних Користувач вводить ім’я, яке вже присутнє, йому виводиться відповідне повідомлення на екран	Якщо при додаванні в базу даних Користувач вводить ім’я, яке вже присутнє, йому виводиться відповідне повідомлення на екран	Пройдено
CF-13	Якщо користувач пробує пройти ідентифікацію при порожній базі даних йому виводиться відповідне повідомлення	Якщо користувач пробує пройти ідентифікацію при порожній базі даних йому виводиться відповідне повідомлення	Пройдено

Продовження таблиці 3.1

CF-14	Користувач не може заносити себе у базу даних не загрузивши файл з голосом або не записавши його онлайн	Користувач не може заносити себе у базу даних не загрузивши файл з голосом або не записавши його онлайн	Пройдено

3.3 Висновки до розділу

При написанні цього розділу було проведено тестування даного продукту та спроба знайти слабкі місця які потенційно були б небезпечні для стійкості та нормальної роботи програми.

Було виявлено, що програма готова до повноцінної експлуатації та відповідає вимогам тестування.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Дане програмне забепечення розроблене для розгортання та функціонування на ОС Linux. Першим кроком до розгортання буде встановлення самої системи Linux.

Послідовність розгортання є наступною:

– розгорнути базу даних MySql на сервері;
– встановити необхідні бібліотеки та пакети на операційну систему;
– запустити дане програмне забезпечення;
– внести додаткових користувачів системи якщо потрібно.

4.2 Робота з програмним забезпеченням

Після розгортання програмного забезпечення воно є готовим до роботи. Першим кроком буде перехід в браузері на посилання http://127.0.0.1:5000. З’явиться вікно роботи з системою. Далі користувачеві буде запропоновано два сценарія роботи з програмою – ідентифіфкація або додання власного голосу в базу даних. Причому в кожному з сценаріїв передбачена або робота з файлом, або з аудіофайлом записаним у реальному часі.

В іншу чергу Адміністратор повинен слідкувати за тим, щоб веб додаток працював та база даних була не порожньою. Для підключення до бази даних та подальшої роботи з нею Адміністратор повинен мати клієнт для підключення до MySql. Також може виникнути ситуація, коли користувач додав себе двічі до бази даних помилково. Отже одним з головних завдань Адміністратора буде комунікація з користувачами системи.

4.3 Висновки до розділу

В розділі 4 описано необхідні умови для розгортання розробленого програмного забезпечення та роботи з ним.

ВИСНОВКИ

Отже, дійсно існує проблема несанкціонованого доступу до даних різних установ – фінансових, юридичних, військових. Одним з найнадійніших підходів до проблеми ідентифікації користувача є використання його біометричних даних – рогівки ока, відбитків пальців, голосу тощо.

Даний проект є чудовим рішенням, бо в першу чергу дає надійний захист системи. Також одним з головних його переваг є простота впровадження – він не потребує сторонніх пристроїв для зчитування вібитків пальців чи ока, користувачеві достатньо мати лише мікрофон.

Під час реалізації даного проекту були використанні новітні підходи в області Науки про дані, Машинного та Глубокого навчання. Зокрема, була використана одна з нейронних мереж сімейства RNN.

Головною перевагою даного проекту також є те, що дану систему можна буде підлаштувати під будь-яку систему ідентифікації будь-якої компанії. Можна обрати іншу базу даних, веб інтерфейс або обрати іншу платформу, переписавши код на C++. Підхід залишиться незмінним.

ПЕРЕЛІК ПОСИЛАНЬ

1) Біометрія / М.П. Горошко, С.І. Миклуш, П.Г. Хомюк. – Львів : Вид-во "Камула" - 2004. – 236 с;

2) Неттер Ф. (переклад Цегельского А.А.) - 2004 - *Атлас анатомії людини*. Львів: Наутілус. с. 592. ISBN 966-95745-8-7;

3) Т. Ганчев, Н. Факотакіс та Г. Кокінакіс (2005), "Порівняльна оцінка різних реалізацій MFCC у завданні перевірки динаміків, заархівована 2011-07-17 на машині Wayback", на 10-й Міжнародній конференції з мови та комп'ютерів (SPECOM 2005) - Вип. 1 - с. 191–194;

4) Клаус Грефф, Рупеш Кумар Срівастава, Ян Кутнік, Bas R. Steunebrink, Юрген Шмідхубер – 2015. «LSTM: пошукова космічна одісея». arXiv: 1503.04069;

5) HTTP [Електронний ресурс] – Режим доступу до ресурсу: http://celnet.ru/vad.php;

6) Ричард Лайонс. Цифровая обработка сигналов. — 2-е. — Москва : Бином - 2006. — 656 с. — ISBN 5-9518-0149-4;

7) HTTP [Електронний ресурс] – Режим доступу до ресурсу: http://eqworld.ipmnet.ru/ru/library/books/Budylin2002ru.pdf;

8) HTTPS [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Speaker_recognition;

9) HTTPS [Електронний ресурс] – Режим доступу до ресурсу: https://dl.acm.org/doi/pdf/10.1145/3194206.3194240?download=true;

10) HTTPS [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА ГОЛОСОМ

Технічне завдання

КПІ.ІП-6322.045420.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Я.П. Опанасенко

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосування для ідентифікації користувача за голосом

Галузь застосування: Будь-які фінансові, державні, юридичні, військові установи**.**

Наведене технічне завдання поширюється на розробку веб-застосування ідентифікації користувача за голосом КПІ.ІП-6322.045420.03.91, котре використовується для ідентифікації користувача використовуючи його біометричну інформацію – голос.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосування для ідентифікації людини за голосом є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (НТУУ «КПІ ім. Сікорського»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для ідентифікації користувача за голосом в системах, де класичні підходи вважаються недостатньо надійними.

Метою створення розробки є покращення надійності систем з обмеженням доступу та ідентифікацією користувачів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– завантаження аудіо файлу;

– можливість вибору сценарію використання – через файл або через запис голосу в реальному часі;

– ідентифікація на основі завантаженого файлу;

– додавання в базу даних на основі завантаженого файлу;

– запис голосу в веб інтерфейсі;

– ідентифікація на основі записаного голосу;

– додавання в базу даних на основі записаного голосу.

4.1.1.2 Для адміністратора:

– створення бази даних користувачів;

– видалення бази даних користувачів;

– видалення користувача з бази даних;

– додавання користувача в базу даних.

4.1.2 Розробка проводилась на платформі Linux. Серверну частину можна розгортати лише на Unix – подібних платформах, але клієнтською версією можна користуватись на будь-якій платформі через браузер.

4.1.3 Додаткові вимоги:

Не передбачені.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

При завантаженні файлів користувач може передавати лише файли “.wav” та “.mp3”.

4.2.2 Забезпечити цілісність інформації в базі даних.

4.2.3 Передбачити захист від некоректних дій користувача.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються.

4.3.2 Обслуговування

Не висуваються.

4.3.3 Обслуговуючий персонал

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на серверах та комп’ютерах з операційною системою Ubuntu/MacOS.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору ……. Pentium.

4.4.2.2 Об‘єм ОЗП ……... 2048 Мб.

4.4.2.3 Об‘єм вільного дискового простору ……. 2048 Мб.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем Ubuntu/MacOS.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі:

.mp3 файл з голосом користувача або запис у реальному часі.

4.5.3 Вихідні дані повинні бути представлені в наступному форматі:

Відображення ідентифікованого користувача на веб-сторінці та коефіцієнт впевненості системи у цьому.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1. Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2. Програмне забезпечення повинно мати довідникову систему

5.3. У склад супроводжувальної документації повинні входити наступні документи:

5.3.1. Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2. Технічне завдання.

5.3.3. Керівництво користувача.

5.3.4. Програма та методика тестування

5.4. Графічна частина повинна бути виконана у формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1. Схема бази даних

5.4.2. Схема структурна класів програмного забезпечення

5.4.3. Креслення вигляду екранних форм

5.4.3. Схема структурна бізнес процесів

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та єтапи розробки наведені у таблиці 6.1.

Таблиця 6.1

№	Назва етапу	Строк	Звітність
1	Вивчення рекомендованої літератури	19.03.2020	
2	Аналіз існуючих методів розв’язання задачі	26.03.2020	
3	Постановка та формалізація задачі	26.03.2020	Технічне завдання
4	Аналіз вимог до програмного забезпечення	02.04.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5	Алгоритмізація задачі	02.04.2020	
6	Моделювання програмного забезпечення	09.04.2020	
7	Обґрунтування використовуваних технічних засобів	16.04.2020	
8	Розробка архітектури програмного забезпечення	23.04.2020	Схема структурна класів програмного забезпечення
9	Розробка програмного забезпечення	30.04.2020	Тексти програмного забезпечення
10	Налагодження програми	07.05.2020	Програма та методика тестування

Продовження таблиці 6.1

11	Виконання графічних документів	14.05.2020	Графічний матеріал проекту
12	Оформлення пояснювальної записки	21.05.2020	Пояснювальна записка проекту
13	Подання ДП на попередній захист	28.05.2020	
14	Подання ДП рецензенту	03.05.2020	
15	Подання ДП на основний захист	08.06.2020	

7 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

7.1. Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА ГОЛОСОМ

Програма та методика тестування

КПІ.ІП-6322.045420.02.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Я.П. Опанасенко

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-застосування для ідентифікації користувача за голосом яке являє собою веб інтерфейс написаний на JavaScript та з’єднано з сервером Flask написаному на Python, який у свою чергу звертається до бази даних MySql.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– функціональна працездатність елементів сторінок web-ресурсу;

– можливість додання користувача в базу даних;

– можливість проходження тесту ідентифікації;

– неможливість додання двох однакових користувачів;

– відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (ба-зове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – спроба завантаження файлу недопустимого розширення, додавання існуючого користувача;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА ГОЛОСОМ

Керівництво користувача

КПІ.ІП-6322.045420.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Я.П. Опанасенко

Київ – 2020 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Проходження тесту ідентифікації

Для проходження тесту ідентифікації користувач повинен обрати один з двох варіантів тестування – за допомогою файлу або записавши свій голос в реальному часі.

При обранні файлу користувач повинен натиснути кнопку «Choose files» та обрати файл для завантаження з локальної файлової системи. Після цього потрібно натиснути кнопку «Identify Speaker».

При обранні тестування за допомогою голосу користувач повинен натиснути кнопку «Record for test (10 sec)» та говорити в мікрофон. Після 10 секунд стане активною кнопка «Identify Speaker». Її треба натиснути.

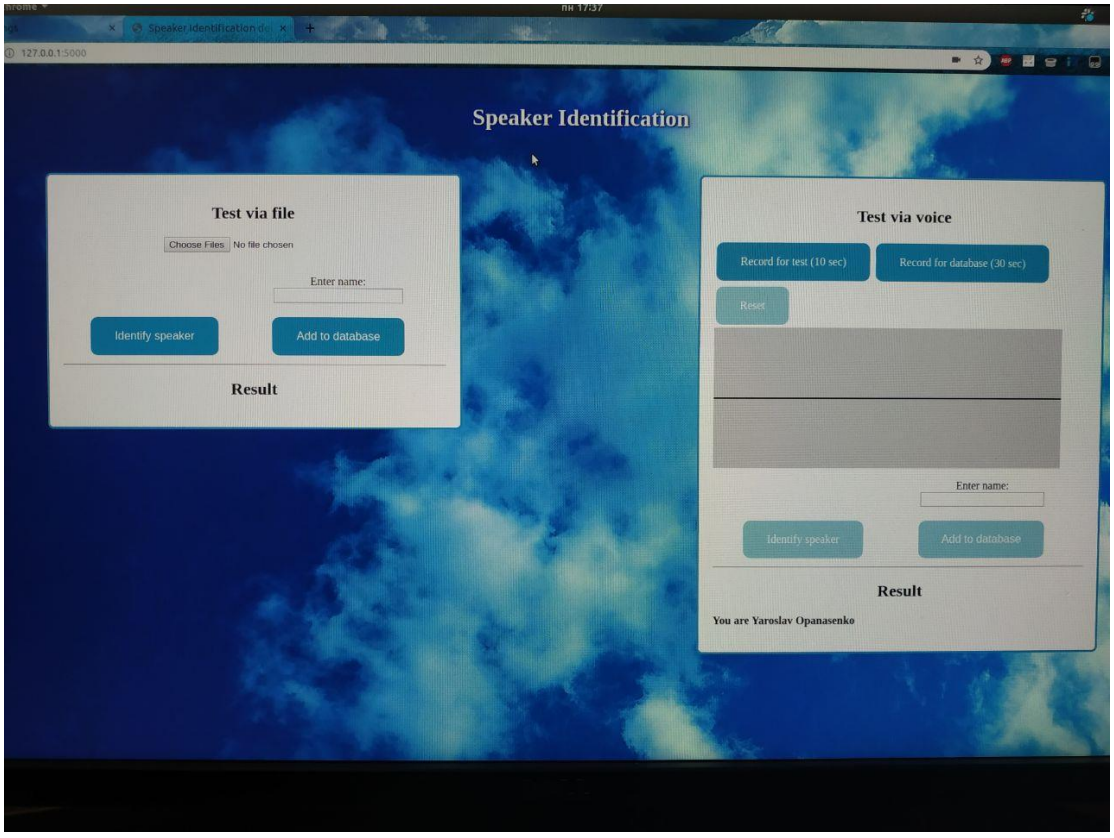

Рисунок 1.1 – Інтерфейс користувача

1.2 Додання користувача в базу даних

Для додання в базу даних користувач повинен обрати один з двох варіантів – за допомогою файлу або записавши свій голос в реальному часі.

При обранні файлу користувач повинен натиснути кнопку «Choose files» та обрати файл для завантаження з локальної файлової системи. Після цього потрібно вказати своє ім’я та натиснути кнопку «Add to database».

При обранні тестування за допомогою голосу користувач повинен натиснути кнопку «Record for database (30 sec)» та говорити в мікрофон. Після 30 секунд стане активною кнопка «Add to database». Треба ввести своє ім’я та натиснути цю кнопку.

2 ІНСТРУКЦІЯ АДМІНІСТРАТОРА

2.1 Створення бази даних користувачів та старт веб застосування

Першим чином Адміністратор повинен підключитися до бази даних через клієнт та створити базу даних. Наступним кроком буде надання прав користувачеві «speech» на цю базу даних. Далі в папці з програмою треба запустити проект командою «python app.py».

ВФакультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА ГОЛОСОМ

Опис програми

КПІ.ІП-6322.045420.06.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Я.П. Опанасенко

Київ – 2020 року

Тексти програмного коду

Веб-застосування для ідентифікації користувача за голосом

(Найменування програми (документа))

DVD-R

(Вид носія даних)

17 арк, 246 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

app.py

```
import os
from flask import Flask, render_template, request, redirect, flash,
jsonify
from identify import identify_front
from identify import append_user_to_db
from flask import abort
from flask import make_response
import json
import numpy as np

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = './data/upload'
app.secret_key = "secret key"
query_list = {}

ALLOWED_EXTENSIONS = set(['wav', 'mp3'])

def allowed_file(filename):
    '''Check if uploaded file is in set of allowed extensions'''
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

@app.route('/')
def upload_form():
    '''Front page of web service'''
    return render_template('upload.html')


@app.route('/test', methods=['GET','POST'])
    def upload_file():
    '''Function of handling verification test from web-service'''
    if request.method == 'POST':
        files = request.files.getlist('file')
        if files[0].filename == 'blob':
            for file in files:
                file.filename = "record.wav"
        if len(files) == 0:
            flash('No files selected for uploading')
            return redirect(request.url)
        if all(allowed_file(file.filename) for file in files):
            os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
#create folder and save files in it
            [file.save(os.path.join(app.config['UPLOAD_FOLDER'],
file.filename)) for file in files]
            flash('Files successfully uploaded')
            file_list = [os.path.join(app.config['UPLOAD_FOLDER'],
file.filename) for file in files]
            if  request.form.getlist('scenario')[0] == 'test':  #
verification test
                time, scores = identify_front(file_list, False) #here
must be browser session id as third argument
                return jsonify({'scores': scores[0][0], 'file':
file_list[0], 'time': time})
            else:                                                #
adding to database
                speaker = request.form.getlist('speaker')[0]
```

```
                append_user_to_db(file_list, speaker)
                return jsonify({'name': speaker})
        else:
            flash('Allowed file types are wav mp3')
            return redirect(request.url)


@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error': 'Not found'}), 404)



#Api part
@app.route('/api/v1.0/identify', methods=['POST'])
def identify_api():
    '''Handling api requests of verification'''
    files = request.files.getlist('file')
    data = json.loads(request.form.getlist('data')[0])
    os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
    [file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
for file in files]
    # query_list[data['query_id']] = 'process'
    file_list = [os.path.join(app.config['UPLOAD_FOLDER'],
file.filename) for file in files]
    time, scores = identify_front(file_list, False, data['query_id'],
data['person_id'], data['single'])    #Take file list, query id, person
id,

# parameter 'single' that means should we

# identificate only this person or top 5
    # query_list[data['query_id']] = 'done'
    users = []
    for sc in scores[:5]:
        users.append({"id": np.int(sc[0]), "similarity": np.round(sc[1],
5)})
    return jsonify(users), 200


@app.route('/api/v1.0/add_to_database', methods=['POST'])
def add_to_db_api():
    '''Handling api requests of adding to database'''
    files = request.files.getlist('file')
    data = json.loads(request.form.getlist('data')[0])
    os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
    [file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
for file in files]
    file_list = [os.path.join(app.config['UPLOAD_FOLDER'],
file.filename) for file in files]
    append_user_to_db(file_list, data['user_id'], data['telephone'])
    return jsonify({'speaker': data['user_id']}), 201
```

```
if __name__ == '__main__':
    app.run(host= '127.0.0.1', debug=True)
```

speech_embedder_net.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  5 20:58:34 2018

@author: harry
"""

import torch
import torch.nn as nn

from hparam import hparam as hp

class (nn.Module):

    def __init__(self):
        super(SpeechEmbedder, self).__init__()
        self.LSTM_stack = nn.LSTM(hp.data.nmels, hp.model.hidden,
num_layers=hp.model.num_layer, batch_first=True)
        for name, param in self.LSTM_stack.named_parameters():
          if 'bias' in name:
             nn.init.constant_(param, 0.0)
          elif 'weight' in name:
             nn.init.xavier_normal_(param)
        self.projection = nn.SpeechEmbeder(hp.model.hidden, hp.model.proj)

    def forward(self, x):
        x, _ = self.LSTM_stack(x.float()) #(batch, frames, n_mels)
        #only use last frame
        x = x[:,x.size(1)-1]
        x = self.projection(x.float())
        x = x / torch.norm(x, p=2, dim=1).unsqueeze(1)
        return x
```

Identify.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import librosa
import numpy as np
import pandas as pd
from sqlalchemy import Table, MetaData
from sqlalchemy import create_engine as create_sql_engine
from sqlalchemy.sql import select
from sqlalchemy.sql import text
# from math import pi
import os
import sys
# import re
import json

#from rnnoise import rnnoise as RN
# import librosa as lr
```

```
from hparam import hparam as hp
from VAD_segments import VAD_chunk_from_bytes
import time

import joblib

import subprocess  # for ffmpeg & embedding_cxx cmd wrapper

    import torch
    from speech_embedder_net import SpeechEmbedder


def get_STFTs(segs):
    if hp.data.mffc:
        return get_STFTs_new(segs)
    else:
        return get_STFTs_old(segs)


def get_STFTs_old(segs):
    #Get 240ms STFT windows with 50% overlap
    sr = hp.data.sr
    STFT_frames = []
    utter_min_len = int(hp.data.window * sr)
    #np.ceil((hp.data.tisv_frame * hp.data.hop + hp.data.window) *
hp.data.sr).astype('int16')
    for seg in segs:
        if len(seg) < utter_min_len:
            continue  # If partial utterance is insufficiently long
        S = librosa.core.stft(y=seg, n_fft=hp.data.nfft,
                              win_length=int(hp.data.window * sr),
hop_length=int(hp.data.hop * sr))
        S = np.abs(S)**2
        mel_basis = librosa.filters.mel(sr, n_fft=hp.data.nfft,
n_mels=hp.data.nmels)
        S = np.log10(np.dot(mel_basis, S) + 1e-6)           # log mel
spectrogram of utterances
        for j in range(0, S.shape[1], int(.12/hp.data.hop)):
            if j + 24 < S.shape[1]:
                STFT_frames.append(S[:,j:j+24])
            else:
                break
    return STFT_frames


def get_STFTs_new(segs):
    # Get 240ms STFT windows with 50% overlap
    sr = hp.data.sr
    STFT_frames = []
    utter_min_len = int(hp.data.window * sr)
    # np.ceil((hp.data.tisv_frame * hp.data.hop + hp.data.window) *
hp.data.sr).astype('int16')
    for seg in segs:
        if len(seg) < utter_min_len:
            continue  # If partial utterance is insufficiently long
        seg_pcm16 = np.round((seg * (2**15 - 1))).astype('int16')
        S = mfcc(seg_pcm16, sr, nfft=hp.data.nfft, winlen=hp.data.window,
winstep=hp.data.hop,
                 numcep=hp.data.nmels, nfilt=hp.data.nmels*2, appendEnergy=True,
                 winfunc=np.bartlett).astype('float32').T
        for j in range(0, S.shape[1], int(.12 / hp.data.hop)):
```

```
            if j + 24 < S.shape[1]:
                STFT_frames.append(S[:, j:j + 24])
            else:
                break
    return STFT_frames


def eprint(*args, **kwargs):
    print(*args, file=sys.stderr, **kwargs)


def pcm16_ffmpeg_enc(sr):
    if sys.byteorder == 'little':
        return ['-f', 's16le', '-acodec', 'pcm_s16le', '-ar', '%d' % sr, '-ac',
'1']
    else:
        return ['-f', 's16be', '-acodec', 'pcm_s16be', '-ar', '%d' % sr, '-ac',
'1']


def ffmpeg_read_pcm16(file, sr):
    '''
    :param file: path to file to load
    :param sr: sample rate to load with
    :return: numpy int16 array of PCM16 data
    '''
    return np.frombuffer(subprocess.run(['ffmpeg', '-v', '0', '-i', file] +
pcm16_ffmpeg_enc(sr) + ['-'],
                         stdout=subprocess.PIPE).stdout, dtype=np.int16)


def ffmpeg_resample_pcm16(audio, sr_old, sr_new):
    '''
    :param audio: numpy int16 array of PCM16 data
    :param sr_old: original sample rate
    :param sr_new: desired sample rate
    :return: numpy int16 array of resampled PCM16 data
    '''
    audio_bytes = audio.tobytes()
    return np.frombuffer(subprocess.run(['ffmpeg', '-v', '0'] +
pcm16_ffmpeg_enc(sr_old) + ['-i', '-'] + \
                          pcm16_ffmpeg_enc(sr_new) + ['-'], input=audio_bytes,
stdout=subprocess.PIPE).stdout,
                         dtype=np.int16)


def ffmpeg_measure_norm(audio_bytes, sr, I=-16, TP=-1.5, LRA=11):
    s = subprocess.run(['ffmpeg', '-v', 'info'] + pcm16_ffmpeg_enc(sr) + ['-i',
'-'] + \
                       ['-af',
'loudnorm=I=%s:dual_mono=true:TP=%s:LRA=%s:print_format=json' % (I, TP, LRA)] +
\
                       ['-f', 'null', '-'], input=audio_bytes,
stderr=subprocess.PIPE).stderr.decode("utf-8")
    json_start = s.rfind('{')
    js = json.loads(s[json_start:])
    return {'I': js['input_i'], 'TP': js['input_tp'], 'LRA': js['input_lra'],
'thresh': js['input_thresh'], 'offset': js['target_offset']}


def ffmpeg_normalize(audio, sr, I=-16, TP=-1.5, LRA=11):
    audio_bytes = audio.tobytes()
```

```
    d = ffmpeg_measure_norm(audio_bytes, sr, I, TP, LRA)
    loudnorm = 'loudnorm=I=%s:TP=%s:LRA=%s' % (I, TP, LRA) + \
      ':measured_I=%s:measured_TP=%s:measured_LRA=%s:measured_thresh=%s' %
(d['I'], d['TP'], d['LRA'], d['thresh']) + \
      ':offset=%s:linear=true' % d['offset']
    return np.frombuffer(subprocess.run(['ffmpeg', '-v', '0'] +
pcm16_ffmpeg_enc(sr) + ['-i', '-'] + \
                                        ['-af', loudnorm] + \
                                        pcm16_ffmpeg_enc(sr) + ['-'],
input=audio_bytes,
                                        stdout=subprocess.PIPE).stdout,
                         dtype=np.int16)



def append_centroid_to_db(centroid, name, telephone = None, con=None,
db_table=None, cl_obj=None):
    if con is None:
        con =
create_sql_engine('mysql+pymysql://%s:%s@localhost/%s?binary_prefix=true'
                                % (hp.database.mysql_user,
hp.database.mysql_password, hp.database.mysql_db))
    if db_table is None:
        db_table = Table('speakers', MetaData(), autoload=True,
autoload_with=con)
    if cl_obj is None:
        cl_obj = joblib.load(hp.database.clusters_obj)
    centroid = centroid / np.linalg.norm(centroid)
    cl = cl_obj.predict(centroid.reshape(1, -1))[0]
    # db_df = pd.DataFrame({'Name': name,
    #                       'Telephone': telephone,
    #                       'Centroid': centroid.astype('float32').tobytes(),
    #                       'ML_Cluster': cl}, [0])
    # db_df.to_sql(name='speakers', con=con, if_exists='append', index=False)
    df = pd.read_sql(text('CALL speakers_insert_proc(:name, :tel, :centroid,
:cl)'),
                     con=con, params={'name': name,
                                      'tel': telephone,
                                      'centroid':
centroid.astype('float32').tobytes(),
                                      'cl': int(cl)},
                     index_col=None)
    if 'ID' in df.columns:
        return df['ID'][0]
    elif 'MYSQL_ERROR' in df.columns:
        raise Exception('MySQL `speakers` table insertion error: %d' %
df['MYSQL_ERROR'][0])
    else:
        return None


def append_user_to_db(files, user_id, telephone = None, con = None):
    append_centroid_to_db(centroid_from_files(files), user_id, telephone, con)

def centroid_from_files(files, model=None, print_times=False):
    if hp.model.use_cxx:
        return np.frombuffer(subprocess.run(['./embedding_cxx'] + files,
                                            stdout=subprocess.PIPE).stdout,
dtype=np.float32)
    else:
        verification_embeddings = []
        if model is None:
```

```
            s_time = time.time()
            model = SpeechEmbedder()
            model.load_state_dict(torch.load(hp.model.model_path))
            model.eval()
            if print_times:
                print("\tmodel loaded: %s" % (time.time() - s_time))
        for file in files:
            s_time = time.time()
            # simulate 8kHz input (for close-to-real testing purpose)
            audio = ffmpeg_read_pcm16(file, 8000)
            audio = ffmpeg_normalize(audio, 8000)
            if hp.data.resample_ffmpeg:
                audio = ffmpeg_resample_pcm16(audio, 8000, 48000)
            else:
                audio = np.array([audio] * 6).T.flatten()  # dirty 8kHz to 48kHz
            if print_times:
                print("\ttranscode done: %s" % (time.time() - s_time))
            s_time = time.time()
            audio = ffmpeg_normalize(audio, 48000)
            # audio = audio[::6]  # 48kHz to 8kHz,
            # audio = ffmpeg_resample_pcm16(audio, 48000, 8000)  # 48kHz to
8kHz,
            byte_audio = audio.tobytes()  # to bytes
            audio = (audio / 32768).astype('float32')  # normalize
            if print_times:
                print("\trnnoise done: %s" % (time.time() - s_time))
            s_time = time.time()
            times, segs = VAD_chunk_from_bytes(2, audio, byte_audio)
            if len(segs) == 0:
                return None, None, None
            # concat_seg = concat_segs(times, segs)
            # STFT_frames = get_STFTs(concat_seg)
            STFT_frames = get_STFTs(segs)
            if len(STFT_frames) == 0:
                return None, None, None
            STFT_frames = np.stack(STFT_frames, axis=2)
            STFT_frames = np.transpose(STFT_frames, axes=(2, 1, 0))
            if print_times:
                print("\tSTFT calculated: %s" % (time.time() - s_time))
            s_time = time.time()
            embeddings = model(torch.from_numpy(STFT_frames)).detach().numpy()
            verification_embeddings.append(embeddings)
        verification_embeddings = np.concatenate(verification_embeddings,
axis=0)
        verification_embedding = np.mean(verification_embeddings, axis=0)
        verification_embedding = verification_embedding /
np.linalg.norm(verification_embedding)
        if print_times:
            print("\tcentroid calculated: %s" % (time.time() - s_time))

    return verification_embedding


def speaker(verification_embedding, con=None, db_table=None,
return_all_scores=True,
            print_times=False, cl_obj=None, cl_thr=0.9, filtered_only=False):
    s_time = time.time()
    if cl_obj is None:
        cl_obj = joblib.load(hp.database.clusters_obj)
    rel_angs = np.array(cl_obj.cluster_centers_).dot(verification_embedding)
    rel_angs = np.arccos(rel_angs) + 1e-8
    rel_angs = (np.min(rel_angs) / rel_angs).flatten()
```

```
    cls = np.argwhere(rel_angs > cl_thr).flatten()
    if print_times:
        print("\tclusters calculated: %s" % (time.time() - s_time))
    if con is None:
        con =
create_sql_engine('mysql+pymysql://%s:%s@localhost/%s?binary_prefix=true'
                                % (hp.database.mysql_user,
hp.database.mysql_password, hp.database.mysql_db))
    if db_table is None:
        db_table = Table('speakers', MetaData(), autoload=True,
autoload_with=con)
    s_time = time.time()
    if filtered_only:
        query = select(
            [db_table.c.ID, db_table.c.Name, db_table.c.Centroid,
db_table.c.ML_Cluster],
            db_table.c.ML_Cluster.in_(cls.tolist())
        )
    else:
        query = select(
            [db_table.c.ID, db_table.c.Name, db_table.c.Centroid,
db_table.c.ML_Cluster]
        )
    db_df = pd.read_sql_query(query, con)
    # db_df['Name'] = db_df['Name'].str.decode('utf-8')

    if print_times:
        print("\tdb records fetched (%d rows): %s" % (db_df.shape[0],
(time.time() - s_time)))
    s_time = time.time()
    db_df['Centroid'] = db_df['Centroid'].apply(lambda x: np.frombuffer(x,
dtype=np.float32).reshape(1, -1))
    if print_times:
        print("\t\tdb centroids unpickled: %s" % (time.time() - s_time))
    s_time = time.time()
    db_centroids = np.concatenate(db_df['Centroid'].values, axis=0)
    db_df['sim'] = db_centroids.dot(verification_embedding)
    if print_times:
        print("\tscores calculated: %s" % (time.time() - s_time))
    if return_all_scores:
        s_time = time.time()
        db_df.sort_values('sim', axis=0, ascending=False, inplace=True)
        if print_times:
            print("\tscores sorted: %s" % (time.time() - s_time))
    else:
        s_time = time.time()
        db_df = db_df.loc[db_df['sim'].argmax(), :]
        if print_times:
            print("\tmax score found: %s" % (time.time() - s_time))
    if print_times:
        print(rel_angs)
    scores = db_df[['ID', 'sim', 'ML_Cluster']].values
    print(type(scores))
    return scores[0][0], scores, cls

def identify(files, filtered_only, con=None):
    db_table = Table('speakers', MetaData(), autoload=True, autoload_with=con)
    for _ in range(1):
        start_time = time.time()
        _, scores, cl = speaker(centroid_from_files(files, print_times=True),
return_all_scores=True,
                                print_times=False, con=con, db_table=db_table,
```

```
                                    filtered_only=filtered_only, cl_thr=0.9)
        cmp_t = time.time() - start_time
        result = 'Pred_classes: ' + str(cl) + 'Scores:'
        for (speaker_, score, cli) in scores[:20]:
            result += '\t%s: [[%s]] class: %s' % (speaker_, np.round(score, 8),
cli)
            print('\t%s: [[%s]] class: %s' % (speaker_, np.round(score, 8),
cli))
        print("--- Compared in %s seconds ---" % (cmp_t))
        return np.round(cmp_t, 3) , scores

def identify_front(file, filtered_only, query_id = '0000000000', person_id = 22,
single = 'False'):
    con =
create_sql_engine('mysql+pymysql://%s:%s@localhost/%s?binary_prefix=true'
                                    % (hp.database.mysql_user,
hp.database.mysql_password, hp.database.mysql_db))
    time, scores = identify(file, filtered_only, con)
    index = np.where(np.transpose(scores)[0] == np.int(person_id))
#select position with appropriate peron id
    if single == 'True':                                                    #if
verification one person
        scores = scores[index[0]]
#select result with appropriate peron id
    else:
        sc = scores[index[0]]
        scores = scores[:5]
        if sc not in scores:
            scores[4] = sc
    # db_df = pd.DataFrame({'query_id': query_id,
    #                       'result': scores.astype('float32').tobytes()}, [0])
    # db_df.to_sql(name='queries', con=con, if_exists='append', index=False)
    df = pd.read_sql(text('CALL queries_insert_proc(:query_id, :result)'),
#Write result into 'queries' table
                     con=con, params={'query_id': query_id,
                                      'result':
scores.astype('float32').tobytes()},
                     index_col=None)
    if 'ID' in df.columns:
#successfully added to 'queries' table
        return time, scores
    elif 'MYSQL_ERROR' in df.columns:
        raise Exception('MySQL `queries` table insertion error: %d' %
df['MYSQL_ERROR'][0])
    else:
        return None

def identify_from_query_id(query_id):
    con =
create_sql_engine('mysql+pymysql://%s:%s@localhost/%s?binary_prefix=true'
                                % (hp.database.mysql_user,
hp.database.mysql_password, hp.database.mysql_db))
    db_table = Table('queries', MetaData(), autoload=True, autoload_with=con)
    query = select(                                                 #Select row with
appropriate query id
        [db_table.c.ID, db_table.c.query_id, db_table.c.result],
        db_table.c.query_id.in_([query_id])
    )
    db_df = pd.read_sql_query(query, con)
    scores = np.frombuffer(db_df['result'][0], dtype=np.float32).reshape(-1, 3)
#Make array from buffer
    return scores
```

```
if __name__ == "__main__":
    if len(sys.argv) not in [2, 3]:
        eprint('Usage: ' + sys.argv[0] + ' <file> [1]')
        eprint('\tWhere\n\t\t<file> is little-endian RIFF 16bit PCM WAVE audio,
mono %d Hz\t\t' % hp.data.sr)
        eprint('\t\t[1] - add optionally "1" to search through all db, not
cluster filtered')
        sys.exit(1)
    file = sys.argv[1]
    if len(sys.argv) == 3 and sys.argv[2] == '1':
        filtered_only = False
    else:
        filtered_only = True
```

record.js

```
import {Fr} from 'jquery.voice.js'

function restore(){
  $("#record, #live").removeClass("disabled");
  $(".one").addClass("disabled");
  Fr.voice.stop();
}
$(document).ready(function(){
  $(document).on("click", "#record:not(.disabled)", function(){
    elem = $(this);
    Fr.voice.record($("#live").is(":checked"), function(){
      elem.addClass("disabled");
      $("#live").addClass("disabled");
      $(".one").removeClass("disabled");
    });
  });

  $(document).on("click", "#stop:not(.disabled)", function(){
    restore();
  });

  $(document).on("click", "#play:not(.disabled)", function(){
    Fr.voice.export(function(url){
      $("#audio").attr("src", url);
      $("#audio")[0].play();
    }, "URL");
    restore();
  });

  $(document).on("click", "#download:not(.disabled)", function(){
    Fr.voice.export(function(url){
      $("<a href='"+url+"' download='MyRecording.wav'></a>")[0].click();
    }, "URL");
    restore();
  });

  $(document).on("click", "#base64:not(.disabled)", function(){
    Fr.voice.export(function(url){
      console.log("Here is the base64 URL : " + url);
      alert("Check the web console for the URL");

      $("<a href='"+ url +"' target='_blank'></a>")[0].click();
    }, "base64");
    restore();
  });
```

```
  $(document).on("click", "#mp3:not(.disabled)", function(){
    alert("The conversion to MP3 will take some time (even 10 minutes), so
please wait....");
    Fr.voice.export(function(url){
      console.log("Here is the MP3 URL : " + url);
      alert("Check the web console for the URL");

      $("<a href='"+ url +"' target='_blank'></a>")[0].click();
    }, "mp3");
    restore();
  });
});
```

test_db.py

```
import os
import glob
import re
import numpy as np
from sqlalchemy import create_engine

from identify import identify
from hparam import hparam as hp

def accuracy():
    engine =
create_engine('mysql+pymysql://%s:%s@localhost/%s?binary_prefix=true'
                                  % (hp.database.mysql_user,
hp.database.mysql_password, hp.database.mysql_db))
    speakers = glob.glob(hp.database.test)
    count = 0
    total = 0
    false_files = []
    for speaker in speakers:
        for f in os.listdir(speaker):       # select file in one speaker
            path_to_file = os.path.join(speaker, f)
            if os.path.isfile(path_to_file) and re.search('.wav$', f):
                total += 1
                time, scores = identify([path_to_file], False, engine)
                if scores[0][0] == os.path.basename(speaker):           # If
verifiacted name equals name of folder
                    count += 1
                else:
                    false_files.append(path_to_file)                    #Append
file to list of false identified files
    print("Part of true identified files -", np.round(count / total, 3))
    print("False files:", false_files)
    return np.round(count / total, 3)

if __name__ == "__main__":
    accuracy()
```

VAD_segments.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 18 16:22:41 2018

@author: Harry
```

```
Modified from https://github.com/wiseman/py-webrtcvad/blob/master/example.py
"""

import collections
import contextlib
import numpy as np
import sys
import librosa
import wave

import webrtcvad

from hparam import hparam as hp

def read_wave(path, sr):
    """Reads a .wav file.
    Takes the path, and returns (PCM audio data, sample rate).
    Assumes sample width == 2
    """
    with contextlib.closing(wave.open(path, 'rb')) as wf:
        num_channels = wf.getnchannels()
        assert num_channels == 1
        sample_width = wf.getsampwidth()
        assert sample_width == 2
        sample_rate = wf.getframerate()
        assert sample_rate in (8000, 16000, 32000, 48000)
        pcm_data = wf.readframes(wf.getnframes())
    data, _ = librosa.load(path, sr)
    assert len(data.shape) == 1
    assert sr in (8000, 16000, 32000, 48000)
    return data, pcm_data

    class Frame(object):
    """Represents a "frame" of audio data."""
    def __init__(self, bytes, timestamp, duration):
        self.bytes = bytes
        self.timestamp = timestamp
        self.duration = duration


def frame_gnerator(frame_duration_ms, audio, sample_rate):
    """Generates audio frames from PCM audio data.
    Takes the desired frame duration in milliseconds, the PCM data, and
    the sample rate.
    Yields Frames of the requested duration.
    """
    n = int(sample_rate * (frame_duration_ms / 1000.0) * 2)
    offset = 0
    timestamp = 0.0
    duration = (float(n) / sample_rate) / 2.0
    while offset + n < len(audio):
        yield Frame(audio[offset:offset + n], timestamp, duration)
        timestamp += duration
        offset += n


def vad_collector(sample_rate, frame_duration_ms,
                  padding_duration_ms, vad, frames):
    """Filters out non-voiced audio frames.
    Given a webrtcvad.Vad and a source of audio frames, yields only
    the voiced audio.
    Uses a padded, sliding window algorithm over the audio frames.
```

```
    When more than 90% of the frames in the window are voiced (as
    reported by the VAD), the collector triggers and begins yielding
    audio frames. Then the collector waits until 90% of the frames in
    the window are unvoiced to detrigger.
    The window is padded at the front and back to provide a small
    amount of silence or the beginnings/endings of speech around the
    voiced frames.
    Arguments:
    sample_rate - The audio sample rate, in Hz.
    frame_duration_ms - The frame duration in milliseconds.
    padding_duration_ms - The amount to pad the window, in milliseconds.
    vad - An instance of webrtcvad.Vad.
    frames - a source of audio frames (sequence or generator).
    Returns: A generator that yields PCM audio data.
    """
    num_padding_frames = int(padding_duration_ms / frame_duration_ms)
    # We use a deque for our sliding window/ring buffer.
    ring_buffer = collections.deque(maxlen=num_padding_frames)
    # We have two states: TRIGGERED and NOTTRIGGERED. We start in the
    # NOTTRIGGERED state.
    triggered = False

    voiced_frames = []
    for frame in frames:
        is_speech = vad.is_speech(frame.bytes, sample_rate)

        if not triggered:
            ring_buffer.append((frame, is_speech))
            num_voiced = len([f for f, speech in ring_buffer if speech])
            # If we're NOTTRIGGERED and more than 90% of the frames in
            # the ring buffer are voiced frames, then enter the
            # TRIGGERED state.
            if num_voiced > 0.9 * ring_buffer.maxlen:
                triggered = True
                start = ring_buffer[0][0].timestamp
                # We want to yield all the audio we see from now until
                # we are NOTTRIGGERED, but we have to start with the
                # audio that's already in the ring buffer.
                for f, s in ring_buffer:
                    voiced_frames.append(f)
                ring_buffer.clear()
        else:
            # We're in the TRIGGERED state, so collect the audio data
            # and add it to the ring buffer.
            voiced_frames.append(frame)
            ring_buffer.append((frame, is_speech))
            num_unvoiced = len([f for f, speech in ring_buffer if not speech])
            # If more than 90% of the frames in the ring buffer are
            # unvoiced, then enter NOTTRIGGERED and yield whatever
            # audio we've collected.
            if num_unvoiced > 0.9 * ring_buffer.maxlen:
                triggered = False
                yield (start, frame.timestamp + frame.duration)
                ring_buffer.clear()
                voiced_frames = []
    # If we have any leftover voiced audio when we run out of input,
    # yield it.
    if voiced_frames:
        yield (start, frame.timestamp + frame.duration)

def VAD_chunk(aggressiveness, path):
    audio, byte_audio = read_wave(path, hp.data.sr)
```

```
    return VAD_chunk_from_bytes(aggressiveness, audio, byte_audio)

def VAD_chunk_from_bytes(aggressiveness, audio=None, byte_audio=None):
    """
    :param aggressiveness: webrtcvad aggressiveness
    :param audio: audio data sequence in librosa style (float32 normalized to [-
1; 1])
    :param byte_audio: same audio in PCM 16bit bytes format
    :return: speech_times, speech_segs
    """
    assert (audio is not None) or (byte_audio is not None)
    if audio is None:
        audio = (np.frombuffer(byte_audio, dtype='int16') /
32768).astype('float32')
    if byte_audio is None:
        byte_audio = np.round(audio * 32767).astype('int16').tobytes()
    vad = webrtcvad.Vad(int(aggressiveness))
    frames = frame_generator(20, byte_audio, hp.data.sr)
    frames = list(frames)
    times = vad_collector(hp.data.sr, 20, 200, vad, frames)
    speech_times = []
    speech_segs = []
    for i, time in enumerate(times):
        start = np.round(time[0],decimals=2)
        end = np.round(time[1],decimals=2)
        old = False
        if old:
            j = start
            while j + .4 < end:
                end_j = np.round(j+.4,decimals=2)
                speech_times.append((j, end_j))

speech_segs.append(audio[int(j*hp.data.sr):int(end_j*hp.data.sr)])
                j = end_j
            else:
                speech_times.append((j, end))
                speech_segs.append(audio[int(j*hp.data.sr):int(end*hp.data.sr)])
        else:
            speech_times.append((start, end))
            speech_segs.append(audio[int(start * hp.data.sr):int(end *
hp.data.sr)])
    return speech_times, speech_segs

if __name__ == '__main__':
    speech_times, speech_segs = VAD_chunk(sys.argv[1], sys.argv[2])
```

hparam.py

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python

import yaml


def load_hparam(filename):
    stream = open(filename, 'r')
    docs = yaml.safe_load_all(stream)
    hparam_dict = dict()
    for doc in docs:
        for k, v in doc.items():
            hparam_dict[k] = v
    return hparam_dict
```

```
def merge_dict(user, default):
    if isinstance(user, dict) and isinstance(default, dict):
        for k, v in default.items():
            if k not in user:
                user[k] = v
            else:
                user[k] = merge_dict(user[k], v)
    return user


class Dotdict(dict):
    """
    a dictionary that supports dot notation
    as well as dictionary access notation
    usage: d = DotDict() or d = DotDict({'val1':'first'})
    set attributes: d.val2 = 'second' or d['val2'] = 'second'
    get attributes: d.val2 or d['val2']
    """
    __getattr__ = dict.__getitem__
    __setattr__ = dict.__setitem__
    __delattr__ = dict.__delitem__

    def __init__(self, dct=None):
        dct = dict() if not dct else dct
        for key, value in dct.items():
            if hasattr(value, 'keys'):
                value = Dotdict(value)
            self[key] = value


class Hparam(Dotdict):

    def __init__(self, file='config.yaml'):
        super(Dotdict, self).__init__()
        hp_dict = load_hparam(file)
        hp_dotdict = Dotdict(hp_dict)
        for k, v in hp_dotdict.items():
            setattr(self, k, v)

    __getattr__ = Dotdict.__getitem__
    __setattr__ = Dotdict.__setitem__
    __delattr__ = Dotdict.__delitem__


hparam = Hparam()
# hparam.training = False
# hparam_test = Hparam()
# hparam_test.training = False
```